massachusetts institute of technology — artificial intelligence laboratory

The Essential Dynamics Algorithm: Essential Results

Martin C. Martin

AI Memo 2003-014 May 2003

Abstract

This paper presents a novel algorithm for learning in a class of stochastic Markov decision processes (MDPs) with continuous state and action spaces that trades speed for accuracy. A transform of the stochastic MDP into a deterministic one is presented which captures the essence of the original dynamics, in a sense made precise. In this transformed MDP, the calculation of values is greatly simplified. The online algorithm estimates the model of the transformed MDP and simultaneously does policy search against it. Bounds on the error of this approximation are proven, and experimental results in a bicycle riding domain are presented. The algorithm learns near optimal policies in orders of magnitude fewer interactions with the stochastic MDP, using less domain knowledge. All code used in the experiments is available on the project's web site.

This work was funded by DARPA as part of the "Natural Tasking of Robots Based on Human Interaction Cues" project under contract number DABT 63-00-C-10102.

1 Introduction

There is currently much interest in the problem of learning in stochastic Markov decision processes (MDPs) with continuous state and action spaces [2, 9, 10]. For such domains, especially when the state or action spaces are of high dimension, the value and Q-functions may be quite complicated and difficult to approximate. However, there may be relatively simple policies which perform well. This has lead to recent interest in *policy search* algorithms, in which the reinforcement signal is used to modify the policy directly [5, 6, 10].

For many problems, a positive reward is only achieved at the end of a task if the agent reaches a "goal" state. For complex problems, the probability that an initial, random policy would reach such a state could be vanishingly small. A widely used methodology to overcome this is *shaping* [1, 3, 4, 8]. Shaping is the introduction of small rewards to reward partial progress toward the goal. A shaping function eases the problem of backing up rewards, since actions are rewarded or punished sooner.

When a policy changes, estimating the resulting change in values can be difficult, requiring the new policy to interact with the MDP for many episodes. In this paper we introduce a method of transforming a stochastic MDP into a deterministic one. Under certain conditions on the original MDP, and given a shaping reward of the proper form, the deterministic MDP can be used to estimate the value of any policy with respect to the original MDP. This leads to an online algorithm for policy search: simultaneously estimate the parameters of a model for the transformed, deterministic MDP, and use this model to estimate both the value of a policy and the gradient of that value with respect to the policy parameters. Then, using these estimates, perform gradient descent search on the policy parameters. Since the transformation captures what is important about the original MDP for planning, we call our method the "essential dynamics" algorithm.

The next section gives an overview of the technique, developing the intuition behind it. In section 3 we describe the mathematical foundations of the algorithm, including bounds on the difference between values in the original and transformed MDPs. Section 4 describes an application of this technique to learning to ride a bicycle. The last section discusses these results, comparing them to previous work. On the bicycle riding task, given the simulator, the only domain knowledge needed is a shaping reward that decreases as lean angle increases, and as angle to goal increases. Compared to previous work on this problem, a near optimal policy is found in *dramatically* less simulated time, and with less domain knowledge.

2 Overview of the Essential Dynamics Algorithm

In the essential dynamics algorithm we learn a model of how state evolves with time, and then use this model to compute the value of the current policy. In addition, if the policy and model are from a parameterized family, we can compute the gradient of the value with respect to the parameters.

In putting this plan into practice, one difficulty is that state transitions are stochastic, so that *expected* rewards must be computed. One way to compute them is to generate many trajectories and average over them, but this can be very time consuming. Instead we might be tempted to estimate only the mean of the state at each future time, and use the rewards associated with that. However, we can do better. If the reward is quadratic, the expected reward is particularly simple. Given knowledge of the state at time t, we can then talk about the distribution of possible states at some later time. For a given distribution of states, let $\bar{s}$ denote the expected state. Then

$$E[r(s)] = \int (a(s-\bar{s})^2 + b(s-\bar{s}) + c)P(s)ds = a\,\mathrm{var}(s) + b(\bar{s}-\bar{s}) + c = a\,\mathrm{var}(s) + c \quad (1)$$

where a, b & c depend on $\bar{s}$.

Suppose the policy depends on a vector of parameters θ. When interacting with the MDP, at every time t after having taken action a_{t-1} in state s_{t-1} and arriving in state s_t:

1. $\tilde{\mu}(s_{t-1}, a_{t-1}) \leftarrow s_t$
2. $\tilde{\nu}(s_{t-1}, a_{t-1}) \leftarrow (s_t - \tilde{\mu}(s_{t-1}, a_{t-1}))^2$
3. $\tilde{s}_t = s_t$
4. $\tilde{\sigma}_t^2 = 0$
5. $\tilde{V} = 0$
6. For every τ in t+1 .. t+n:
   a. $\tilde{s}_\tau = \tilde{\mu}(\tilde{s}_{\tau-1}, \pi(\tilde{s}_{\tau-1}))$
   b. $\tilde{\sigma}_\tau^2 = \tilde{\nu}(\tilde{s}_{\tau-1}, \pi(\tilde{s}_{\tau-1})) + \tilde{\sigma}_{\tau-1}^2(\tilde{\mu}'_\pi(\tilde{s}_{\tau-1}))$
   c. $\tilde{r}_\tau = \frac{1}{2} r''(\tilde{s}_\tau)\tilde{\sigma}_\tau^2 + r(\tilde{s}_\tau)$
   d. $\tilde{V} = \tilde{V} + \gamma^{\tau-t}\tilde{r}_\tau$
7. Update the policy in the direction that increases $\tilde{V}$: $\theta = \theta + \alpha\frac{\partial \tilde{V}}{\partial \theta}$

Figure 1: The essential dynamics algorithm for a one dimensional state space. The notation $f(x) \leftarrow a$ means "adjust the parameters that determine f to make $f(x)$ closer to a," e.g. by gradient descent. $\tilde{\mu}'_\pi$ is the derivative of $\tilde{\mu}(s, \pi(s))$ with respect to s.

Thus, to calculate the expected reward, we don't need to know the full state distribution, but simply its mean and variance. Thus, our model should describe how the mean and variance evolve over time. If the state transitions are "smooth," they can be approximated by a Taylor series. Let π be the current policy, and let $\mu_\pi(s)$ denote the expected state that results from taking action $\pi(s)$ in state s. If $\bar{s}_t$ denotes the mean state at time t, and σ_t^2 the variance, and if state transitions were deterministic, then to first order we would have

$$\bar{s}_{t+1} \approx \mu_\pi(\bar{s}_t)$$
$$\sigma_{t+1}^2 \approx \left(\frac{d\mu_\pi}{ds}(\bar{s}_t)\right)^2 \sigma_t^2$$

where μ_π' is the derivative of μ_π with respect to state. For stochastic state transitions, let $\nu_\pi(s)$ be the variance of the state that results from taking action $\pi(s)$ in state s. It turns out that the variance at the next time step is simply $\nu_\pi(s)$ plus the transformed variance from above, leading to

$$\bar{s}_{t+1} \approx \mu_\pi(\bar{s}_t)$$
$$\sigma_{t+1}^2 \approx \nu_\pi(\bar{s}_t) + \left(\frac{d\mu_\pi}{ds}(\bar{s}_t)\right)^2 \sigma_t^2 \tag{2}$$

Thus, we learn estimates $\tilde{\mu}$ and $\tilde{\nu}$ of μ and ν respectively, use Eq. (2) to estimate the mean and variance of future states, and Eq. (1) to calculate the expected reward. The resulting algorithm, which we call the *expected dynamics* algorithm, is presented in Figure 1.

The next section gives a formal derivation of the algorithm, and proves error bounds on the estimated state, variance, reward and value for the general n-dimensional case, where the reward is only approximately quadratic.

3 Derivation of the Essential Dynamics Algorithm

A Markov Decision Process (MDP) is a tuple $\langle S, D, A, P_{s,a}, r, \gamma \rangle$ where: S is a set of *states;* $D: S \to \mathbb{R}$ is the *initial-state distribution;* A is a set of *actions;* $P_{s,a}: S \to \mathbb{R}$ are the *transition probabilities;* $r: S \times A \to \mathbb{R}$ is the *reward;* and γ is the *discount factor.* This paper is concerned with continuous state and action spaces, in particular we assume $S = \mathbb{R}^{n_s}$ and $A = \mathbb{R}^{n_a}$. We use subscripts to denote time and superscripts to denote components of vectors and matrices. Thus, s_t^i denotes the ith component of the vector s at time t.

A (deterministic) *policy* is a mapping from a state to the action to be taken in that state, $\pi: S \to A$. Given a policy and a distribution P_t of states at time t, such as the initial state distribution or the observed state, the distribution of states at future times is defined by the recursive relation $P_{\tau+1}(s) = \int_S P_{s',\pi(s')}(s) P_\tau(s') ds'$ for $\tau > t$. Given such a distribution, we can define the expectation and the covariance matrix of a random vector x with respect to it, which we denote $\mathrm{E}_t[x]$ and $\mathrm{cov}_t(x)$ respectively. Thus, $\mathrm{E}_t[x] = \int x P_t(x) dx$ and $\mathrm{cov}_t^{i,j}(x) = \mathrm{E}_t[(x^i - \mathrm{E}_t[x^i])(x^j - \mathrm{E}_t[x^j])]$. When P_t is zero except for a single state s_t, we introduce $\mathrm{E}[x|s_t]$ as a synonym for $\mathrm{E}_t[x]$ which makes the distribution explicit.

Given an MDP, we define the *limited horizon value function* for a given policy as $V_\pi(s_t) = \sum_{\tau=t}^{n} \gamma^{\tau-t} \mathrm{E}_\tau[r(s_\tau, \pi(s_\tau))]$ where the probability density at time t is zero except for state s_t. Also given a policy, we define two functions, the mean $\mu_\pi(s)$ and covariance matrix $\nu_\pi(s)$ of the next state. Thus, $\mu_\pi(s_t) = \mathrm{E}[s_{t+1}|s_t]$ and $\nu_\pi(s_t) = \mathrm{E}[(s_{t+1} - \mu_\pi(s_t))(s_{t+1} - \mu_\pi(s_t))^T | s_t]$. In *policy search,* we have a fixed set of policies Π and we try to find one that results in a value function with high values.

We transform the stochastic MDP M to a deterministic one $M' = \langle S', s_0', A', f', r', \gamma \rangle$ as follows. A state in the new MDP is an ordered pair consisting of a state from S and a covariance matrix, denoted (s, Σ). The new initial state $s_0' = (\mathrm{E}_D[s], \mathrm{cov}_D[s])$. The new action space is the set of all possible policies for M, that is $A' = \{\pi | \pi: A \to S\}$. The state transition probabilities are replaced with a (deterministic) *state transition function* $f'(s', a')$, which gives the unique successor state that results from taking action $a'_t = \pi$ in state $s'_t = (s_t, \Sigma_t)$. We set $f'(s'_t, a'_t) = f'(s_t, \Sigma_t, \pi) = (\mu_\pi(s_t), \nu_\pi(s_t) + (\nabla\mu_\pi)\Sigma_t(\nabla\mu_\pi)^T)$. The reward $r'(s, \Sigma, \pi) = r(s) + \frac{1}{2}\mathrm{tr}\left(\left[\frac{\partial^2 r}{\partial i \partial j}(s)\right]\Sigma\right)$ where $\left[\frac{\partial^2 r}{\partial i \partial j}(s)\right]$ denotes the matrix of second derivatives of r with respect to each state variable. Finally, $\gamma' = \gamma$.

The strength of the method comes from the theorems below, which state that the above transform approximately captures the dynamics of the original probabilistic MDP to the extent that the original dynamics are "smooth." The first theorem bounds the error in approximating state, the second in covariance, the third in reward and the fourth in value.

Theorem 1 *Fix a time t, a policy π, and a distribution of states P_t. Choose M_μ and M*

such that $\forall s, \sqrt{\sum_{i,j,k=1}^{n_s}\left(\frac{\partial^2}{\partial j \partial k}\mu_\pi^i(s)\right)^2} < M_\mu$, $\|\nabla\mu_\pi(\tilde{s}_t)\| < M$ *and* $\|\mathrm{cov}_t(s_t, s_t)\|_F < M$, *where* $\|\ \|_F$ *denotes the Frobenius norm. Let* $\tilde{s}_t$ *be given, and define* $\tilde{s}_{t+1} = \mu_\pi(\tilde{s}_t)$, $\varepsilon_t^s = \mathrm{E}_t[s_t] - \tilde{s}_t$ *and* $\varepsilon_{t+1}^s = \mathrm{E}_t[s_{t+1}] - \tilde{s}_{t+1}$. *Then* $\|\varepsilon_{t+1}^s\| < (\|\varepsilon_t^s\| + M_\mu)\left(\frac{3}{2}M + \frac{1}{2}\|\varepsilon_t^s\|^2\right)$.

Theorem 2 *Suppose* M_ν *and* M *are chosen so that* $\forall s$, $\sqrt{\sum_{i,j,k=1}^{n_s}\left(\frac{\partial \nu^{i,j}}{\partial k}(s)\right)^2} < M_\nu$, $\left\|\mathrm{E}_t[|s_t - \mathrm{E}_t[s_t]|^k]\right\|_F < M$ *for* $k = 1, 2, 3, 4$, $\|\tilde{s}_{t+1}\| = \|\mu_\pi(\tilde{s}_t)\| < M$ *and all the conditions of Theorem 1. Let* $\tilde{\Sigma}_t$ *be given, and define* $\tilde{\Sigma}_{t+1}^{i,j} = \nu^{i,j}(\tilde{s}_t) + (\nabla\mu^i(\tilde{s}_t))^T\tilde{\Sigma}_t(\nabla\mu^j(\tilde{s}_t))$. *Let* $\varepsilon_t^\Sigma = \mathrm{cov}_t(s_t, s_t) - \tilde{\Sigma}_t$, *similarly for* ε_{t+1}^Σ. *Then*

$$\|\varepsilon_{t+1}^\Sigma\|_F \le (\|\varepsilon_t^\Sigma\|_F + \|\varepsilon_t^s\| + M_\mu + M_\nu)M^2(10 + O(\|\varepsilon_t^s\|))$$

Theorem 3 *Suppose* $\forall s, \sqrt{\sum_{i,j,k=1}^{n_s}\left(\frac{\partial^3 r}{\partial i \partial j \partial k}(s)\right)^2} < M_r$, $\sqrt{\sum_{i,j=1}^{n_s}\left(\frac{\partial^2 r}{\partial i \partial j}(\tilde{s}_t)\right)^2} < M$ *and* $\|\nabla r(\tilde{s}_t)\| < M$ *and the conditions of the previous two theorems. Let* $\varepsilon_t^r = \mathrm{E}_t[r(s_t)] - r'(\tilde{s}_t)$. *Then* $\mathrm{E}_t[r(s_t)] = r'(\tilde{s}_t) + \varepsilon_t^r = r(\tilde{s}_t) + \frac{1}{2}\mathrm{tr}\left(\left[\frac{\partial^2 r}{\partial i \partial j}\right]\tilde{\Sigma}_t\right) + \varepsilon_t^r$ *where*

$$|\varepsilon_t^r| < (\|\varepsilon_t^\Sigma\|_F + \|\varepsilon_t^s\| + M_r)\left(\frac{5}{3}M + O(\|\varepsilon_t^s\|)\right)$$

Theorem 4 *Fix a time* t *and a policy* π, *and a distribution of states* P_t. *Let* $\tilde{s}_t$ *and* $\tilde{\Sigma}_t$ *be given, and define* $\tilde{s}_\tau$ *and* $\tilde{\Sigma}_\tau$ *for* $\tau = t+1 \ldots t+n$ *recursively as in theorems 1 and 2 above. Let* $M_{\varepsilon r}$ *be an upper bound for* $|\varepsilon_\tau^r|$ *for all* $\tau \in [t, t+n]$. *Then under the conditions of the above three theorems,* $\mathrm{E}[V(s_t)] = V'(\tilde{s}_t) + \varepsilon_t^V$ *where* $\varepsilon_t^V < \frac{1-\gamma^{n+1}}{1-\gamma}M_{\varepsilon r}$.

Proof: First, some preliminaries. In the first three theorems, which deal only with a single transition and a single distribution of states at time t, namely P_t, let $\bar{x} = \mathrm{E}_{P_t}[x]$ for any random variable x. Note that for any vector x and square matrices A and B, $x^TAx = \mathrm{tr}(A(xx^T))$ where tr(.) denotes the trace of a matrix, $|\mathrm{tr}(AB)| \le \|A\|_F\|B\|_F$, and $\|xx^T\|_F = |x|^2$. In the statement of theorem 2, $\mathrm{E}_t[|(s_t - \bar{s}_t)^3|]$ is a three dimensional matrix whose i, j, k element is $\mathrm{E}_t[|(s_t^i - \bar{s}_t^i)(s_t^j - \bar{s}_t^j)(s_t^k - \bar{s}_t^k)|]$. Similarly, $\mathrm{E}_t[|(s_t - \bar{s}_t)^4|]$ is a four dimensional matrix, and if all of its elements are finite, then the lower powers must also be finite. The Frobenius norm of such matrices is simply the square root of the sum of the squares of all their elements. Also, if a, b, c & d are real numbers that are greater than zero, then $ab + cd < (a+c)(b+d)$.

Note that, since μ_π is a vector valued function, $\nabla\mu_\pi(s)$ is a matrix. Since μ_π^i, the ith component of μ_π, is a real valued function, $\nabla\mu_\pi^i(s) \in \mathbb{R}^{n_s}$. Because $\nu(s)$ is a matrix, $\nu^{i,j}(s) \in \mathbb{R}$. Let $\left[\frac{\partial^2}{\partial j \partial k}\mu_\pi^i(x)\right]$ denote the matrix of second partial derivatives of μ_π^i, evaluated at $x \in \mathbb{R}^{n_s}$. For any s, let $\Delta_1 = s - \bar{s}_t$, $\Delta_2 = \bar{s}_t - \tilde{s}_t$ and $\Delta = \Delta_1 + \Delta_2 = s - \tilde{s}_t$.

Thus, $E_{P_t}[\Delta] = \Delta_2$ and $E_t[\Delta\Delta^T] = E_t[\Delta_1\Delta_1^T] + \Delta_2\Delta_2^T = \Sigma_t + \Delta_2\Delta_2^T = \tilde{\Sigma}_t + \varepsilon_t^{\Sigma} + \Delta_2\Delta_2^T$. Note that $\Delta_2 = \varepsilon_t^s$.

Proof of Theorem 1: Expand $\mu_\pi^i(s)$ using a first order Taylor series with the Lagrange form of the remainder, namely

$$\mu_\pi^i(s) = \mu_\pi^i(\tilde{s}_t) + \nabla\mu_\pi^i(\tilde{s}_t)^T(s-\tilde{s}) + \frac{1}{2}(s-\tilde{s}_t)^T\left[\frac{\partial^2}{\partial j\partial k}\mu_\pi^i(x)\right](s-\tilde{s}_t), \tag{3}$$

i.e.

$$\mu_\pi^i(s) = \mu_\pi^i(\tilde{s}_t) + \nabla\mu_\pi^i(\tilde{s}_t)^T\Delta + \frac{1}{2}\Delta^T\left[\frac{\partial^2}{\partial j\partial k}\mu_\pi^i(x)\right]\Delta \tag{4}$$

for some x on the line joining s and $\tilde{s}_t$. Then

$$\begin{aligned} E_{P_t}[s_{t+1}^i] - \tilde{s}_{t+1}^i &= E_{P_t}[\mu_\pi^i(s_t)] - \mu_\pi^i(\tilde{s}_t) \\ &= \mu_\pi^i(\tilde{s}_t) + \nabla\mu_\pi^i(\tilde{s}_t)^T\Delta_2 + \frac{1}{2}\mathrm{tr}\left(\left[\frac{\partial^2}{\partial j\partial k}\mu_\pi^i(x)\right](\Sigma_t + \Delta_2\Delta_2^T)\right) - \mu_\pi^i(\tilde{s}_t) \end{aligned}$$

So $\|\varepsilon_{t+1}^s\| < M\|\varepsilon_t^s\| + \frac{1}{2}M_\mu(M + \|\varepsilon_t^s\|^2) < (\|\varepsilon_t^s\| + M_\mu)\left(M + \frac{1}{2}(M + \|\varepsilon_t^s\|^2)\right)$. ■

Proof of Theorem 2: Let $M_k' = \left\|E_t[|(s_t - \bar{s}_t)^k|]\right\|_F$. By the mean value theorem, $\nu^{i,j}(s) = \nu^{i,j}(\bar{s}_t) + \nabla\nu^{i,j}(x)\cdot\Delta$ for some x on the line joining s and $\tilde{s}_t$. Also, $\nu^{i,j}(s_t) = E[s_{t+1}^i s_{t+1}^j | s_t] - \mu^i(s_t)\mu^j(s_t)$ so that

$$\begin{aligned} \mathrm{cov}_{P_t}(s_{t+1}^i, s_{t+1}^j) &= E[s_{t+1}^i s_{t+1}^j] - \overline{s_{t+1}^i}\,\overline{s_{t+1}^j} \\ &= E_{P_t}[E[s_{t+1}^i s_{t+1}^j | s_t]] - \overline{s_{t+1}^i}\,\overline{s_{t+1}^j} \\ &= \nu^{i,j}(\bar{s}_t) + E_{P_t}[\nabla\nu^{i,j}(x)\cdot\Delta] + E_{P_t}[\mu^i(s_t)\mu^j(s_t)] - \overline{s_{t+1}^i}\,\overline{s_{t+1}^j} \end{aligned} \tag{5}$$

The second term is an error term, call it $\varepsilon'^{i,j}$. We have $|\varepsilon'| < M_\nu M_1'$. For the third term, we expand both μ^i and μ^j using Eq. (4) and multiplying out the terms, obtaining

$$\begin{aligned} E_{P_t}[\mu^i(s_t)\mu^j(s_t)] &= \mu^i(\tilde{s}_t)\mu^j(\tilde{s}_t) \\ &+ \mu^i(\tilde{s}_t)\nabla\mu^j(\tilde{s}_t)^T\Delta_2 + \mu^j(\tilde{s}_t)\nabla\mu^i(\tilde{s}_t)^T\Delta_2 \\ &+ \nabla\mu^i(\tilde{s}_t)^T(\tilde{\Sigma}_t + \varepsilon_t^{\Sigma} + \Delta_2\Delta_2^T)\nabla\mu^j(\tilde{s}_t) \\ &+ \frac{1}{2}\mu^i(\tilde{s}_t)^T E_{P_t}\left[\Delta^T\left[\frac{\partial^2}{\partial k\partial l}\mu_\pi^j(x)\right]\Delta\right] + \frac{1}{2}\mu^j(\tilde{s}_t)^T E_{P_t}\left[\Delta^T\left[\frac{\partial^2}{\partial k\partial l}\mu_\pi^i(x)\right]\Delta\right] \\ &+ \frac{1}{2}\nabla\mu^i(\tilde{s}_t)^T E_{P_t}\left[\Delta\Delta^T\left[\frac{\partial^2}{\partial k\partial l}\mu_\pi^j(x)\right]\Delta\right] + \frac{1}{2}\nabla\mu^j(\tilde{s}_t)^T E_{P_t}\left[\Delta\Delta^T\left[\frac{\partial^2}{\partial k\partial l}\mu_\pi^i(x)\right]\Delta\right] \\ &+ \frac{1}{4}E_{P_t}\left[\Delta^T\left[\frac{\partial^2}{\partial k\partial l}\mu_\pi^i(x)\right]\Delta\Delta^T\left[\frac{\partial^2}{\partial k\partial l}\mu_\pi^j(x)\right]\Delta\right] \end{aligned}$$

All terms other than the first and the one involving $\tilde{\Sigma}_t$ are error terms, call their sum $\varepsilon''^{i,j}$. That is,

$$E_{P_t}[\mu^i(s_t)\mu^j(s_t)] = \mu^i(\tilde{s}_t)\mu^j(\tilde{s}_t) + \nabla\mu^i(\tilde{s}_t)^T\tilde{\Sigma}_t\nabla\mu^j(\tilde{s}_t) + \varepsilon''^{i,j}$$

where

$$\|\varepsilon''\| < 2\|\tilde{s}_{t+1}\|\|\nabla\mu(\tilde{s}_t)\|\|\varepsilon_t^s\| + \|\nabla\mu(\tilde{s}_t)\|^2(\|\varepsilon_t^s\|^2 + \|\varepsilon_t^\Sigma\|_F)$$
$$+ \|\tilde{s}_{t+1}\|M_\mu M_2' + \|\nabla\mu(\tilde{s}_t)\|M_\mu M_3' + \frac{1}{4}M_\mu^2 M_4'$$

Lastly let $\varepsilon_2''' = \mu^i(\tilde{s}_t)\mu^j(\tilde{s}_t) - \overline{s_{t+1}^i}\,\overline{s_{t+1}^j}$. By Theorem 1,

$$\begin{aligned}\varepsilon'''^{i,j} &= \mu^i(\tilde{s}_t)\mu^j(\tilde{s}_t) - \overline{s_{t+1}^i}\,\overline{s_{t+1}^j} \\ &= \mu^i(\tilde{s}_t)\mu^j(\tilde{s}_t) - (\mu^i(\tilde{s}_t) + \varepsilon_t^{si})(\mu^j(\tilde{s}_t) + \varepsilon_t^{sj}) \\ &= -\mu^i(\tilde{s}_t)\varepsilon_t^{si} - \mu^j(\tilde{s}_t)\varepsilon_t^{si} - \varepsilon_t^{si}\varepsilon_t^{sj}\end{aligned}$$

$$\|\varepsilon'''\| < 2\|\varepsilon_t^s\|\|\tilde{s}_{t+1}\| + \|\varepsilon_t^s\|^2$$

Substituting into Eq. (5), we obtain:

$$\mathrm{cov}_{P_t}(s_{t+1}^i, s_{t+1}^j) = \nu^{i,j}(\tilde{s}_t) + \varepsilon'^{i,j} + \nabla\mu^i(\tilde{s}_t)^T\tilde{\Sigma}_t\nabla\mu^j(\tilde{s}_t) + \varepsilon''^{i,j} + \varepsilon'''^{i,j}$$

so that $\varepsilon_{t+1}^\Sigma = \varepsilon' + \varepsilon'' + \varepsilon'''$ and

$$\|\varepsilon_{t+1}^\Sigma\|_F < M_\nu M_1' + 2M^2\|\varepsilon_t^s\| + M^2(\|\varepsilon_t^s\|^2 + \|\varepsilon_t^\Sigma\|) + MM_\mu M_2' + MM_\mu M_3' + \frac{1}{4}M_\mu^2 M_4'$$
$$+ 2(\|\varepsilon_t^s\| + M_\mu)\left(\frac{3}{2}M + \frac{1}{2}\|\varepsilon_t^s\|^2\right)M + (\|\varepsilon_t^s\| + M_\mu)^2\left(\frac{3}{2}M + \frac{1}{2}\|\varepsilon_t^s\|^2\right)^2$$

Each term has at least one of the "small bounds" $\|\varepsilon_t^s\|$, $\|\varepsilon_t^\Sigma\|_F$, M_μ or M_ν. Using the inequality from the preliminaries, we can "factor them out." The four M_k' are bounded by $M + O(\|\varepsilon_t^s\|)$, as can be shown using the binomial theorem, e.g.

$$\begin{aligned}E_t[|\Delta_1 + \Delta_2|^3] &\le E_t[(|\Delta_1| + |\Delta_2|)^3] \\ &= E_t[|\Delta_1|^3] + 3|\Delta_2|E_t[|\Delta_1|^2] + 3|\Delta_2|^2E_t[|\Delta_1|] + |\Delta_2|^3 \\ &= E_t[|\Delta_1|^3] + O(\|\varepsilon_t^s\|)\end{aligned}$$

. ■

Proof of Theorem 3: Expand $r(s)$ using a second order Taylor series with the Lagrange form of the remainder, namely

$$r(s) = r(\tilde{s}_t) + \nabla r(\tilde{s}_t)^T\cdot\Delta + \frac{1}{2}\Delta^T\left[\frac{\partial^2 r}{\partial i\partial j}(\bar{s}_t)\right]\Delta + \frac{1}{6}\sum_{i,j,k=1}^{n_s}\frac{\partial^3 r}{\partial i\partial j\partial k}(x)\Delta^i\Delta^j\Delta^k. \qquad (6)$$

Call the last term ε'. Thus,

$$\begin{aligned}E_t[r(s)] &= r(\tilde{s}_t) + \nabla r(\tilde{s}_t)^T\cdot\Delta_2 + \frac{1}{2}\mathrm{tr}\left(\left[\frac{\partial^2 r}{\partial i\partial j}(\bar{s}_t)\right](\tilde{\Sigma}_t + \varepsilon_t^\Sigma + \Delta_2\Delta_2^T)\right) + E_t[\varepsilon'] \\ &= r'(\tilde{s}_t) + \varepsilon_t^r\end{aligned}$$

and

$$\begin{aligned}|\varepsilon_t^r| &< \|\nabla r(\tilde{s}_t)\|\|\varepsilon_t^s\| + \frac{1}{2}(\|\varepsilon_t^\Sigma\|_F + \|\varepsilon_t^s\|^2)M + \frac{1}{6}M_r M_3' \\ &< (\|\varepsilon_t^s\| + \|\varepsilon_t^\Sigma\|_F + M_r)\left(M + \frac{1}{2}M + \frac{1}{2}M\|\varepsilon_t^s\| + \frac{1}{6}M_3'\right)\end{aligned}$$

. ■

Proof of Theorem 4:

$$\begin{aligned}
\mathrm{E}[V(s_t)] &= \sum_{\tau=t}^{n} \gamma^{\tau-t} \mathrm{E}_\tau[r(s_\tau)] \\
&= \sum_{\tau=t}^{n} \gamma^{\tau-t} (r'(\tilde{s}_\tau) + \varepsilon_\tau^r) \\
&= V'(\tilde{s}_t) + \sum_{\tau=t}^{n} \gamma^{\tau-t} \varepsilon_\tau^r
\end{aligned}$$

So,

$$\left|\varepsilon_t^V\right| < \sum_{\tau=t}^{n} \gamma^{\tau-t} M_{\varepsilon r} = M_{\varepsilon r} \sum_{\tau=t}^{n} \gamma^{\tau-t} = M_{\varepsilon r} \frac{1-\gamma^{n+1}}{1-\gamma} . \blacksquare$$

The above theorems state that as long as $\left\|\varepsilon_t^{\dot{s}}\right\|$, $\left\|\varepsilon_t^{\Sigma}\right\|_F$, M_μ, M_ν and M_r are small and M is finite, and given a good estimate of the mean and covariance of the state at some time, the transformed MDP will result in good estimates at later times, and hence the reward and value functions will also be good estimates. Note that no particular distribution of states is assumed, only that, essentially, the first four moments are bounded at every time. The most unusual conditions are that the reward r be roughly quadratic, and that the value function include only a limited number of future rewards. This motivates the use of shaping rewards.

4 Experiments

The code used for all experiments in this paper is available from www.metahuman.org/martin/Research.html.

The essential dynamics algorithm was applied to Randløv and Alstrøm's bicycle riding task [8], with the objective of riding a bicycle to a goal 1km away. The five state variables were simply the lean angle, the handlebar angle, their time derivatives, and the angle to the goal. The two actions were the torque to apply to the handlebars and the horizontal displacement of the rider's center of mass from the bicycle's center line. The stochasticity of state transitions came from a uniform random number added to the rider's displacement. If the lean angle exceeded $\pi/15$, the bicycle fell over and the run terminated.

If the variance of the state is not too large at every time step, then the variance term in the transformed reward can simply be considered another form of error, and only $\tilde{\mu}$ need be estimated. This was done here. A continuous time formulation was used where, instead of estimating the values of the state variables at a next time, their derivatives were estimated. The model was of the form

$$\frac{\partial s^i}{\partial t} = \tilde{\mu}_{w^i}(s, a) = w^i \cdot \varphi(s, a)$$

where $\varphi(s, a)$ was a vector of features and w^i was a vector of weights. The features were simply the state and action variables themselves. The derivative of each state variable was estimated using gradient descent on w_i with the error measure $err_i = \left|\dot{s}^i - w_i \cdot \varphi(s, a)\right|$ and a learning rate of 1.0. This error measure was found to work better than the more traditional squared error. The squared error is minimized by the mean of the observed values, whereas the absolute value is minimized by the median [7]. The median is a more robust estimate of central tendency, i.e. less susceptible to outliers, and therefore may be a better choice in many practical situations.

Model estimation was done online, simultaneous with policy search. In the continuous formulation, the value function is the time integral of the reward times the discount factor. The future state was estimated using Euler integration [7]. While the bicycle simulator also used Euler integration, these choices were unrelated. In fact, $\Delta t = 0.01$s for the bicycle simulator and 0.051s for integrating the estimated reward. It was integrated for 30 time steps.

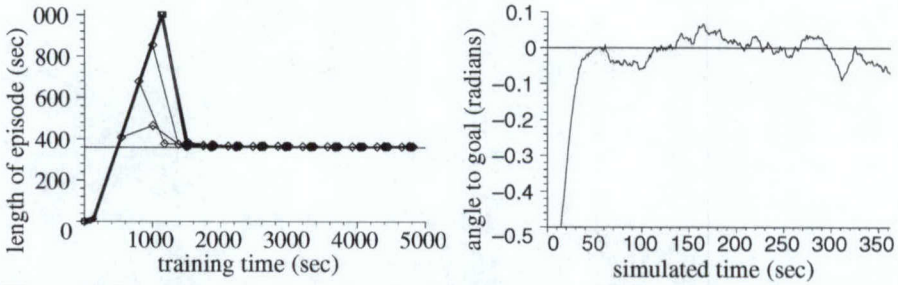

Figure 2: The left graph shows length of episode vs. training time for 10 runs. The dashed line indicates the optimal policy. Stable riding was achieved within 200 simulated seconds. The right graph shows angle to goal vs. time for a single episode starting after 3000 simulated seconds of training.

The shaping reward was the square of the angle to goal plus 10 times the square of the lean angle. The policy was a weighted sum of features, with a small Gaussian added for exploration, $\pi(s) = \theta \cdot \varphi(s) + N(0, 0.05)$. The features were simply the state variables themselves. When the model is poor or the policy parameters are far from a local optimum, $\partial V/\partial\theta$ can be quite large, resulting in a large gradient descent step which may overshoot its region of applicability. This can be addressed by reducing the learning rate, but then learning becomes interminably slow. Thus, the gradient descent rule was modified to $\frac{\partial\theta}{\partial t} = -\alpha\frac{\partial V/\partial\theta}{(\beta + \|\partial V/\partial\theta\|)}$. Near an optimum, when $\|\partial V/\partial\theta\| \ll \beta$, this reduces to the usual rule with a learning rate of α/β. In this experiment, $\alpha = 0.01$ and $\beta = 1.0$.

A graph of episode time vs. learning time is shown in Figure 1. After falling over between 40 and 60 times, the controller was able to ride to the goal or the time limit without falling over. After a single such episode, it consistently rode directly to the goal in a near minimum amount of time. The resulting policy was essentially an optimal policy.

5 Discussion

For learning and planning in complex worlds with continuous, high dimensional state and action spaces, the goal is not so much to converge on a perfect solution, but to find a good solution within a reasonable time. Such problems often use a shaping reward to accelerate learning. For a large class of such problems, this paper proposes approximating the problem's dynamics in such a way that the mean and covariance of the future state can be estimated from the observed current state. We have shown that, under certain conditions, the rewards in the approximate MDP are close to those in the original, with an error that grows boundedly as time increases. Thus, if the rewards are only summed for a limited number of steps ahead, the resulting values will approximate the values of the original system. Learning in this transformed problem is considerably easier than in the original, and both model estimation and policy search can be achieved online.

The simulation of bicycle riding is a good example of a problem where the value function is complex and hard to approximate, yet simple policies produce near optimal solutions. Using a traditional value function approximation approach, Randløv needed to augment the state with the second derivative of the lean angle ($\ddot{\omega}$) and provide shaping rewards [8]. The resulting algorithm took 1700 episodes to ride stably, and 4200 episodes to get to the goal for the first time. The resulting policies tended to ride in circles and precess toward the goal, riding roughly 7 km to get to a goal 1 km away.

In contrast, when the action is a weighted sum of (very simple) features, random search can find near optimal policies. This was tested experimentally; 0.55% of random policies consistently reached the goal when $\ddot{\omega}$ was included in the state, and 0.30% did

when it wasn't.[1] What's more, over half of these policies had a path length within 1% of the best reported solutions. Policies that rode stably but not to the goal were obtained 0.89% and 0.24% of the time respectively. Thus, a random search of policies needs only a few hundred episodes to find a near optimal policy.

The essential dynamics algorithm consistently finds such near optimal policies, and the author is aware of only one other algorithm which does, the PEGASUS algorithm of [5]. The experiments in this paper took 40 to 60 episodes to ride stably, that is, to the goal or until the time limit without falling over. After a single such episode, the policy consistently rode directly to the goal in a near minimum amount of time. In contrast, PEGASUS used at least 450 episodes to evaluate each policy.[2] One reasonable initial policy is to always apply zero torque to the handlebars and zero displacement of body position. This falls over in an average of 1.74 seconds, so PEGASUS would need 780 simulated seconds to evaluate such a policy. The essential dynamics algorithm learns to ride stably in approximately 200 simulated seconds, and in the second 780 simulated seconds will have found a near optimal policy.

This was achieved using very little domain knowledge. $\ddot{\Omega}$ was not needed in the state, and the features were trivial. The essential dynamics algorithm can be used for online learning, or can learn from trajectories provided by other policies, that is, it can "learn by watching." In the bicycle experiment, the essential dynamics algorithm needed many times more computing power per simulated second than PEGASUS, although it was still faster than real time on a 1GHz mobile Pentium III, and therefore could presumably be used for learning on a real bicycle. The experiments in section 4 added the square of the lean angle to the shaping reward, but did not use any information about dynamics (i.e. velocities or accelerations), nor about the handlebars. In fact, the shaping reward simply corresponded to the common sense advice "stay upright and head toward the goal."

However, these advantages do not come without drawbacks. The essential dynamics algorithm only does policy search in an approximation to the original MDP, so an optimal policy for this approximate MDP won't, in general, be optimal for the original MDP. The theorems in section 3 give bounds on this error, and for bicycle riding this error is small.

Conclusion

This paper has presented an algorithm for online policy search in MDPs with continuous state and action spaces. A stochastic MDP is transformed to a deterministic MDP which captures the essential dynamics of the original. Policy search is then be performed in this transformed MDP. Error bounds were given and the technique was applied to a simulation of bicycle riding. The algorithm found near optimal solutions with less domain knowledge and orders of magnitude less time than existing techniques.

Acknowledgements

The author would like Leslie Kaelbling, Ali Rahimi and especially Kevin Murphy for enlightening comments and discussions of this work.

---

1. Our experiment contained two conditions, namely with or without $\ddot{\Omega}$ in the state, resulting in 5 or 6 state variables. The features were the state variables themselves, state and action variables were scaled to roughly the range [-1, +1], weights were chosen uniformly from [-2, +2], and each policy was run 30 times. In 100,000 policies per condition, 549 (0.55%) reached the goal all 30 times when $\ddot{\Omega}$ was included, and 300 (0.30%) when it wasn't. For such policies, the median riding distance was 1009m and 1008m respectively. The code used is available on the web site.

2. [5] evaluated a given policy by simulating it 30 times. The derivative with respect to each of the 15 weights was evaluated using finite differences, requiring another 30 simulations per weight, for a total of 30×15 = 450 simulations. Often, the starting weights at a given stage were evaluated during the previous stage, so only the derivatives need to be calculated.

References

[1] Colombetti, M. & Dorigo, M. (1994) Training agents to perform sequential behavior. In *Adaptive Behavior,* **2**(3), pp. 247-275.

[2] Forbes, J., & Andre, D. (2000) Real-time reinforcement learning in continuous domains. In *AAAI Spring Symposium on Real-Time Autonomous Systems.*

[3] Mataric, M.J. (1994) Reward functions for accelerated learning. In W.W. Cohen and H. Hirsch (eds.) *Proc. 11th Intl. Conf. on Machine Learning.*

[4] Ng, A. et al. (1999) Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. 16th Intl. Conf. on Machine Learning*, pp. 278-287.

[5] Ng, A. & Jordan, M. (2000) PEGASUS: A policy search method for large MDPs and POMDPs. In *Uncertainty in Artificial Intelligence (UAI), Proc. of the Sixteenth Conf.*, pp. 406-415.

[6] Peshkin, L. et al. (2000) Learning to Cooperate via Policy Search. In *Uncertainty in Artificial Intelligence (UAI), Proc. of the Sixteenth Conf.*, pp. 307-314.

[7] Press, W. H. et al. (1992) *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press.

[8] Randløv, J. (2000) Shaping in Reinforcement Learning by Changing the Physics of the Problem. In *Proc. Intl. Conf. on Machine Learning*. pp. 767-774.

[9] Santamaría, J.C. et al. (1998) Experiments with Reinforcement Learning in Problems with Continuous State and Action Spaces. In *Adaptive Behavior,* **6**(2), 1998

[10] Strens, M. J. A. & Moore, A.W. (2002) Policy Search using Paired Comparisons. In *Journal of Machine Learning Research,* v. 3, pp. 921-950.